ISI/SR-76-7
June 1976

ARPA ORDER NO. 2223

PROTECTION ERRORS IN OPERATING SYSTEMS:

Allocation/Deallocation Residuals

Dennis Hollingworth

Richard Bisbey II

INFORMATION SCIENCES INSTITUTE

UNIVERSITY OF SOUTHERN CALIFORNIA

4676 Admiralty Way/Marina del Rey/California 90291
(213) 822-1511

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ISI/SR-76-7
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Protection Errors in Operating Systems: Allocation/Deallocation Residuals.
5. TYPE OF REPORT & PERIOD COVERED: Research Rept.
6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Dennis Hollingworth, Richard Bisbey, II
8. CONTRACT OR GRANT NUMBER(s): DAHC 15-72-C0308, ARPA Order-2223
9. PERFORMING ORGANIZATION NAME AND ADDRESS: USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90291
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA Order #2223, Program Code 3D30 & 3P10
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209
12. REPORT DATE: June 1976
13. NUMBER OF PAGES: 17 (18p.)
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): ----
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document is approved for public release and sale; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

-----

18. SUPPLEMENTARY NOTES

-----

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Residuals, computer security, operating systems, protection evaluation, protection policy

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A common security problem is the residual--data or access capability left after the completion of a process and not intended for use outside the context of that process. If the residual becomes accessible to another process, a security error may result. A major source of such residuals is improper or incomplete allocation/deallocation processing. The various types of allocation/deallocation residuals are discussed in terms of their characteristics and the manner in which they occur, and a semiautomatable search strategy for detecting sources of these residuals is presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED 407952

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ISI/SR-76-7
June 1976

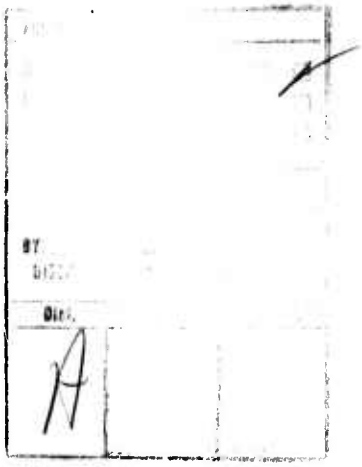
ARPA ORDER NO. 2223

PROTECTION ERRORS IN OPERATING SYSTEMS:

Allocation/Deallocation Residuals

Dennis Hollingworth

Richard Bisbey II



INFORMATION SCIENCES INSTITUTE

UNIVERSITY OF SOUTHERN CALIFORNIA

4676 Admiralty Way/Marina del Rey/California 90291
(213) 822-1511

THIS RESEARCH IS SUPPORTED BY THE ADVANCED RESEARCH PROJECTS AGENCY UNDER CONTRACT NO. DAHC15 72 C 0308, ARPA ORDER NO. 2223 PROGRAM CODE NO. 3D30 AND 3P10.

ABSTRACT

A common security problem is the residual--data or access capability left after the completion of a process and not intended for use outside the context of that process. If the residual becomes accessible to another process, a security error may result. A major source of such residuals is improper or incomplete allocation/deallocation processing. The various types of allocation/deallocation residuals are discussed in terms of their characteristics and the manner in which they occur, and a semiautomatable search strategy for detecting sources of these residuals is presented.

PREFACE

This document is one of a series of related reports, each describing a specific type of security problem found in contemporary computer operating systems and suggesting techniques for finding errors of that type in a variety of systems (different versions, manufacturers, etc.). The reports are intended for use by persons responsible for evaluation and enhancement of the security of existing operating system software. These studies will assist individuals having no particular expertise in the field of operating system security to effectively carry out these tasks.

The particular security problem addressed by this document is that of the potential for security violations resulting from failure of deallocation/allocation procedures to completely destroy all residue of the previous use of a storage object. Such "residuals" constitute a widely recognized form of security problem which warrants treatment in its own right. While it is recognized that security errors involving residuals can occur in a variety of ways, this document is restricted to those which are associated with improper or incomplete allocation/deallocation processing.

ACKNOWLEDGMENT

Many of the ideas in this report were clarified, and their presentation improved, as a result of suggestions by our collaborator on the Protection Analysis Project, Jim Carlstedt.

INTRODUCTION: GENERAL CHARACTERISTICS

Intuitively, the notion of a residual suggests something which is left over upon the completion of a process (or any discrete computational sequence) but not intended for use outside the context of that process. A residual may be (1) data directly generated or utilized by the process, (2) data indicative of the computational activity of the process, or (3) accessibility to a cell* no longer associated with the process. The following are examples of residuals:

1. Data on a scratch tape which is not destroyed when the tape is deallocated.

2. Data in main memory (such as a system message buffer) which is not destroyed when the memory (buffer) is released by a process.

3. System working data in a user-supplied buffer, generated by a supervisor routine as a side effect of servicing a request, which is not destroyed by the supervisor prior to its relinquishing control.

4. The contents of a CPU register which are not purged when the processor is switched between processes.

5. Information intrinsic to the cell's structural composition, indicative of its previous use, which is not adequately destroyed through cell decomposition.

6. Data in archive storage accessible to a newly registered user of the system as a result of his being assigned the same identifier or user number as a previous user.

7. Access capability which still exists after the denoted cell has been deallocated.

A residual error occurs when the residual becomes accessible to another process. In Examples 1, 2, and 4 the cell is subject to allocation to another process; consequently, any information associated with the cell potentially becomes accessible to any process in execution. If access to that information by the recipient process violates some security policy, the residual error constitutes a security error.

In order to recognize residual errors it is necessary to distinguish between legitimate communications and residual data and to recognize changes in cell accessibility; both are difficult in the general case. Example 3 above illustrates the first difficulty: the

---

* By "cell" we refer to any container of information which gives identity to a collection of information and hence may be viewed as a single entity. Cells can exist at various levels of structure, both logical and physical. For example, a file is a cell which may be viewed as a discrete container of information or as an aggregation of other cells, known as records, which in turn may be composed of still other cells, known as fields, etc. Alternatively, that same file can be viewed as being composed of physical tracks, which are cells composed of physical records, which are cells composed of words of disk memory, which are also cells.

work buffer used as a communications vehicle between the supervisor routine and the invoking process may, in the event of abnormal termination, contain system information, constituting a residual error. Even in the case of normal termination the buffer may contain residual information in addition to the intended communication. As an example of the latter difficulty, code which allows or disallows user asynchronous I/O activity can in effect change cell accessibility and thus be as important as code which changes an access control list or a page-table.

A situation in which residuals can be distinguished and changes in cell accessibility recognized is the case of resource "allocation/deallocation"; this document is restricted to residual errors arising from such processing.

CELL ALLOCATION/DEALLOCATION PROCESSING

Inasmuch as this report addresses residuals associated with errors in the allocation/deallocation process, it is useful to examine such processes to gain insight into how and where residual errors may occur. In this document, the term "allocation (deallocation)" will apply to any action on behalf of a process to acquire (release) accessibility to a cell, whether or not that action is accomplished by code formally labeled as such (for example, the unbinding of a virtual memory page from a real memory page is an instance of deallocation). Intrinsic to this notion is the concept of a "free-pool," a set to which a cell no longer reserved by any process is returned for reuse.

No single model of cell allocation/deallocation encompasses the wide variety of forms assumed by the process in contemporary operating systems. Different models are required in order to reflect different areas of functionality within the same system and even similiar areas of functionality across different systems. However, certain events can be identified which are associated with the generation of residual errors.

Both the allocation and deallocation processes involve two activities: access management and cell management. Access management refers to the creation or destruction of access-paths to a cell, where an access-path is a (reference-path,operator) pair, resulting in the enabling or disabling of a process from applying particular operators along indicated reference-paths. (See the section on access management residuals.) Cell management refers to the acquisition or disposition of the cell itself.

Allocation Processing

In allocation processing, cell management consists of finding a named cell or selecting from a free-pool a cell of a specified type (possibly composing it from other cells). It may also include initialization specific to the cell type (e.g., setting a time stamp in a message buffer, formatting disk tracks of a new file, supplying skeletal information for a control block, etc.).

Another function of cell management is to establish a "usage reservation" on the cell being allocated. Each cell must undergo some change or be marked in some way so as to prevent simultaneous allocation of nonsharable cells and account for concurrent usage of sharable cells. For nonsharable cells this may only require logically removing the cell from the free-pool in which it resides. For a sharable cell, the usage reservation generally includes a usage-indicator which reflects, in part, the number or identity of the users of the cell.

The access management task involves the creation of an access-path to the cell being allocated, i.e., establishing a reference-path (e.g., a page-table entry, a segment-table entry, a directory link) from the name-space of the requesting process to the cell and enabling the path for a specified set of process-invocable operators. Reference-path creation may not be meaningful for all types of cells in all systems (for example, all of main memory in IBM's System/360 is intrinsically addressable by any process).

Deallocation Processing

Deallocation, the converse of allocation, consists of partial or complete destruction of cell accessibility for the process, followed by cell management processing. The former involves the disabling of a process from applying a specified set of operators along a specific set of reference-paths and may also involve the destruction of the reference-paths themselves. Residual accessibility to the cell can result from failure to identify and destroy all relevant access paths or failure to disable a process from applying a particular operator along a specific path. When no more reservations on the cell are extant, deallocation processing proceeds with cell management processing.

Cell management processing involves disposition of the deallocated cell, which may be retained for later use with its identity and content intact or released to a free-pool. In the latter case, it might be decomposed (possibly recursively) into more primitive cells before being introduced into the appropriate free-pool(s). (For the purpose of this report, the decision process as to whether a released cell is decomposed or not is of little interest; it may depend upon a variety of criteria with respect to resource usage demands, expected requirement for cells of this type, etc.) In addition to disposing of cells, the cell management task is responsible for destroying cell attributes (e.g., cell content, size, structure) which might otherwise be transmitted through the free-pool by the deallocation/allocation process and constitute residual errors. This is typically done when the cell is joined to the free-pool. The decision of which attributes to destroy depends largely on (1) the security policy which applies to the cell (e.g., sensitivity of the content), (2) any specific action regarding residuals explicitly or implicitly indicated by the deallocation request, and (3) any global security policy in effect for residuals in general.

Having briefly sketched the basic functions involved in allocation/deallocation, the remainder of this report focuses on the various types of residual errors. Each is discussed in the context of the functional area in allocation/deallocation in which it occurs; a search procedure for finding the sources of each type of residual error is included. Two basic residual types are those resulting from errors in cell management (attribute residuals) and access management (access residuals).

CELL MANAGEMENT RESIDUALS

As seen above, a basic function of the cell management task is to dispose of the deallocated cell. A particularly important aspect of this activity (from a security standpoint) is the proper handling of any residual associated with the usage of the storage cell by the deallocating process. The residual can take the form of any attribute of the cell which is preserved through deallocation and subsequent reallocation. Numerous attributes can be addressed, including content, size, time of last use, location, structure, etc. The specific processing required with respect to such attribute residuals depends largely on the enforcement technique. In this report, content residuals will be treated separately from those involving other attributes.

Not all attribute residuals represent security violations; a distinction must be made between those which may be allowed and those which must be prevented. Considerable latitude in the choice of enforcement policies is possible, ranging from the prevention of all attribute residuals to the prevention of only those which are also security errors. Considerations such as run-time efficiency, functional simplicity, "precautionary" security measures, and the requirement for data confinement [Lampson 1973] may be involved in the choice of a particular enforcement policy. For example, to increase run-time efficiency, a cell might be cleared only if the process to which the cell is being allocated is different from the process from which the cell is being deallocated, or if it contains data not normally accessible to the allocating process (i.e., only if the residual error can constitute a security error). Alternatively, the acceptability of some run-time inefficiency together with the desire for functional simplicity or precautionary security measures might result in an enforcement policy stating that all content residuals be destroyed upon deallocation of a cell, regardless of the processes involved or the content of the cell. Likewise, the desire to increase data confinement by eliminating potentially high bandwidth communication channels necessitates that certain attribute residuals be destroyed.

Enforcement policy must be chosen within the constraints imposed by the capabilities of the existing hardware protection mechanisms. For example, a chosen enforcement policy might state that newly allocated cells must be written or cleared before being read. However, the resolution of the protection mechanism may be insufficient to support the requisite cell and access mode discrimination. Systems such as IBM's System/360, for example, can enforce storage protection only on 2048-byte blocks, while Honeywell's 6180 can enforce storage protection only on a per-segment basis. Similarly, granting access in one mode may imply granting access in other modes (e.g., write accessibility may imply read accessibility), and in some cases the two are not distinguished. Thus, granting access in any form may require that the cell or even an entire block or segment has already been cleared and initialized.

Content Residuals

An obvious example of a content residual involves memory or file space which has been allocated to a process without all data from the previous allocation having been purged. The operating system penetration attack generally referred to as "scavenging" exploits conditions of this type. The penetration routine requests an allocation of storage

space and reads the storage prior to writing it, perusing it for noninitialized cells which contain sensitive data left there from a previous process.

Search Strategy. The possibility of diverse enforcement policies and deallocation strategies precludes the identification of a specific scheme for identifying the sources of content residual errors. However, it is possible to outline a somewhat general approach, the majority of which is manual because of the unavailability of a suitably discriminating recognition algorithm, but aspects of which are amenable to automation.

Systematic identification of the sources of content residual errors in an operating system requires identifying for each type of cell the corresponding allocation/deallocation code. The process starts with identifying all cell types defined for the target system, an essentially manual activity requiring detailed knowledge of the subject operating system, since it requires recognizing not only simple cell types but also complex cell types defined across media or with noncontiguous parts.

One way in which identifying cell types may be facilitated is to first identify the physical media and storage units with which residual errors might be associated; they include magnetic or paper tape, hardware device buffers, channel or CPU registers, cards, laser store, and the like. Identification of the media in which cells (and hence, residual errors) may exist makes it possible to identify the various types of cells which are in whole or in part allocated and deallocated in each medium. For disk storage these might include single-medium cells such as the volume-table-of-contents, directory entries, file records, unused space records, index records, end-of-file indicators, overflow and linkage records, disk labels, password files, etc. The cell might also be a component of a multi-media cell (suggesting the existence of the more abstract cell types). For example, the primary index for an index-sequential file may exist in main memory while secondary indices and the file itself exist on disk.

Another way to identify cell types is to examine the data declarations in the system source listings, which might be located in the source listings manually or via an automated data declaration recognizer and then analyzed to determine the cell types which they represent.

Having identified the various cell types, the evaluator must manually identify the free-pools which serve as buffers for cell resources between deallocation and allocation. The free-pools in turn are the basis for identifying allocation and deallocation code. Associated with each free-pool are one or more control variables (pool headers, counters, etc.) used in the insertion and extraction of cell elements. References to these variables indicate free-pool manipulation, and hence allocation and deallocation code. An automated global symbol search may be used to identify all instructions which reference the control variables. For each identified point, the surrounding code must be examined to determine if the reference to the variable involves insertion or extraction of elements of the free-pool, corresponding to deallocation and allocation, respectively.

In summary, deallocation and allocation code can be located by (1) identifying cell types, (2) for each cell type identifying its particular free-pool(s), and (3) for each free-pool, identifying the insertion/extraction code and hence the deallocation/allocation

code. When the deallocation and allocation code has been identified, the remaining task is to establish that the handling of residuals is consistent with the enforcement policy.

The content of a deallocated cell may be destroyed at one of two points: (1) before the cell is added to the free-pool or (2) after the the cell has been extracted from the free-pool (the latter is a less desirable situation from a security standpoint, since a functional error elsewhere in the system may lead to exposure of the residual while it exists in the free-pool). If the content is destroyed when the cell is added to the free-pool, two sets of control paths leading to the free-pool insertion point(s) must be distinguished: those for which content residue is ostensibly destroyed and those for which it is not. For the former set it is necessary to verify that the entire residual is destroyed.

Conditionals which result in a path being in one set or the other must be consistent with the desired enforcement policy; this may or may not be readily determinable, depending largely upon the complexity of the enforcement policy and the associated code. For example, if the policy states that all content residue associated with cells of a given type be destroyed, then it is necessary to ensure only that all paths leading to the free-pool insertion point are in the first set. However, if destruction of content residue is a user option, then it must be verified that election of that option results in an appropriate path being taken.

If the content residual is destroyed after the elements are extracted from the free-pool, an analogous set of considerations apply. In this case, the paths under consideration are those emanating from the extraction point. If there is a conditional which must be evaluated at allocation, relevant information about the deallocating process or the cell's previous use (e.g., classification, previous owner, etc.) may have to be preserved for the allocation process. The decisions made based on that information must be consistent with the desired enforcement policy.

Residuals Involving Other Attributes

Residual information can be transmitted through the free-pool via cell attributes other than the content attribute. Like the content attribute, preservation of these attributes through the deallocation/allocation process may (1) establish a hidden communication channel which can be exploited by cooperating processes to defeat attempts at data containment, or (2) allow information about the process deallocating the cell to be deduced by other processes. [Lampson 1973] Two attributes seem particularly important: cell size, and inter- and intra-cell relationship.

Knowledge of the size and use of a previously allocated cell can often provide security-relevant information. For example, the fact that a cell used as a password buffer is N characters long implies that the previous password was N or less characters long. This information might significantly reduce the number trials necessary to guess the password. If the size of a deallocated cell is either directly determinable by a user process or the cell itself is reconstructed in response to an allocation request of nonspecific size, then a size residual results. Size residuals typically appear in systems with free-pool management schemes in which the cell is retained intact in the free-pool

(without being decomposed into smaller cells or assimilated into larger blocks), and allocations are serviced via a best-fit strategy.

The relationship between cells as preserved in the free-pool (i.e., their location, order in the free-pool, etc.) may also convey security-relevant information. The cells may be subcells of a previously deallocated cell and thus give information about the cell's structure, or they may be individually deallocated cells and convey information about the activity of a specific process. For example, the existence in the free-pool of a set of cells in a particular size sequence may suggest their previous use (e.g., as elements of a particular type of control block), and hence the previous activity of the deallocating process. Likewise, the order in the free-pool of a series of deallocated buffers may suggest the structure of a larger process-defined cell such as a file record. Inter-/intra-cell relationship residuals typically appear in systems with free-pool management schemes that employ order-preserving insertion/extraction algorithms such as last-in/first-out.

Search Strategy. Much of the approach employed in identifying sources of content residuals is applicable to identifying sources of residuals involving other attributes; in fact, the two are identical through the point of locating that code involved with insertion and extraction of elements from the free-pool. As with content residuals, the allocation/deallocation logic must be examined to ensure that the attribute in question is destroyed. To prevent size residuals, the decomposition code must be analyzed to ensure that the size of the cell is not preserved in a recognizable way when it is introduced into the free-pool. To prevent preservation of inter-/intra-cell relationship, the insertion and extraction algorithms must be analyzed to ensure that the pair is not order-preserving.

ACCESS MANAGEMENT RESIDUALS

The second major source of allocation/deallocation residuals is the access management function. The primary function of the access management task is to create (destroy) access-paths as indicated in the allocation (deallocation) request such that specific operators may (no longer) be applied to the specified cell along specific paths by the allocating (deallocating) process. If not done properly, process accessibility to the cell may still exist after the deallocation or be inadvertently established as a side-effect of the allocation. Such a condition is called an "access residual."

Before we proceed with an analysis of the possible forms of access residuals, it is instructive to develop a model of the name translation mechanism to illustrate those factors which must be considered in reference- path management.

Name Translation

A process acts on a cell via a reference-path, which may be thought of as a triple of the form (name,context,translation-mechanism). The elements of the triple are related via the functional relationship

T(name,context) = physical address of cell

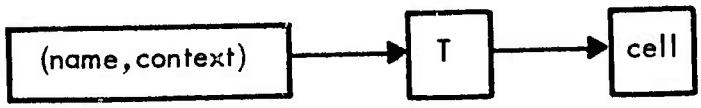
where "T" is the translation mechanism, "name" is the identifier by which the cell is denoted in the process-name-space (e.g., logical unit number, virtual memory address), and "context" is the supporting data which participates in the name translation process (e.g., a segment table). The process references the cell by implicitly (through instruction execution or procedure invocation) invoking the translation-mechanism and either implicitly (through operator selection) or explicitly supplying to it the name and context.

The translation process may be single-step (e.g., addition of the base register value in a base-relocation scheme) or arbitrarily complex, involving a number of translation steps representing different levels of translation:

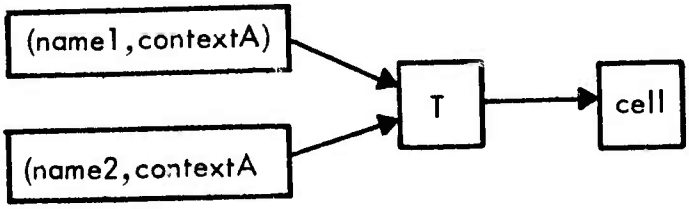
T(name,context) = tn(fn(name),...,t2(f2(name),t1(f1(name),context))...)

Ignoring indirection, each level of translation corresponds to a level of cell composition with segments of the name being used in separate steps of the translation process (f is a subname selector, and t is a context selector). Name translation may be be broken or interrupted at a variety of points (e.g., segment number interpretation, page number interpretation). Indirection may further complicate the name interpretation process by requiring successive application of the above-described process to successive results until the final cell address is determined.

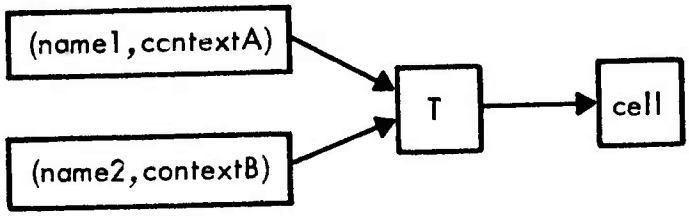
A number of reference-path configurations can exist for a given cell: in the simplest case, a single reference-path exists to the given cell (Figure 1a); within a given context a cell may have been identified by several names (Figure 1b); different processes may access the same cell by different reference-paths (Figures 1c and 1d).



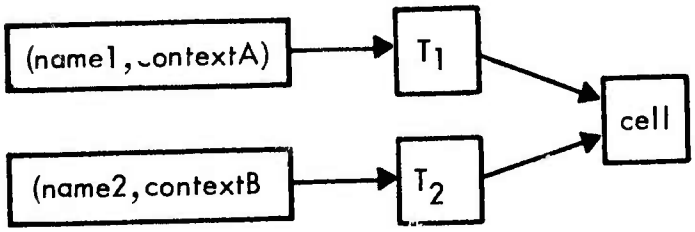
(a)



(b)



(c)



(d)

Figure 1.

Associated with a given reference-path is a mechanism which either permits or prevents a process from applying an operator to the cell via the reference-path on the basis of a capability, a storage-key, a bounds register, or the like. In some systems the mediation mechanism and the reference-path mechanism are intertwined; in other systems the two are separate.

Access Residuals

An access residual is an access-path not destroyed despite a deallocation request specifying that access-path. All access residuals constitute residual errors. Destruction of an access-path may consist of either deleting a reference-path or disabling a process from applying a particular operator along that path. (The latter might entail as little as setting a bit from zero to one.)

Access residuals result by means of the following:

1. Failure to disable either all of the indicated operators associated with the specified access-path (e.g., cell writability is to be disabled but the relevant indicator is not reset) or the identified reference-path in response to a deallocation request.

2. Failure to recognize and hence process all access-paths relevant to a particular deallocation request.

Case 2 merits further discussion. As was indicated in the name translation model above, multiple access-paths may exist to a cell either through separate cell names (e.g., multiple allocations of the same cell by the same process), separate contexts (e.g., allocation of the same cell by different processes), different translation mechanisms (e.g., different procedures), or combinations thereof. Such access-paths may be created not only in response to formal allocation requests, but also as a consequence of operators which copy existing reference-path data or use the reference-path mechanism to translate a name and then store the result for later use. For example, an I/O or message reply request may be fully translated to the physical address of the cell and then stored for future use. Likewise, for lengthy translations, efficiency considerations may dictate that the same translation process not be repeated for every use of that name. All such created access-paths must be properly accounted for.

Search Strategy. Identification of sources of access-residual errors is complicated by the following two problems:

1 Although manifested at cell deallocation, access residuals may be the result of functional errors at cell allocation (i.e., a specification mismatch between allocation and deallocation code in accounting for allocated cells). For example, access-paths may be established during cell allocation processing which are unknown to the deallocation process and consequently not deleted.

2. The code which produces, destroys, or copies stored, translated references as well as the references themselves may be scattered throughout the operating system, making both difficult to locate and analyze. The stored, translated names may not even be referencable by (and hence locatable through) the cell management code. For example, cell management processing might know only the number, but not the location, of outstanding resolved references with cell management processing suspended until the number is zero.

Detection of sources of access residuals involves two phases: identification of access management code, and evaluation of that code.

The first phase is a multistep process, portions of which are identical to that used in detecting attribute residuals. Access management code falls into two categories: that which is associated with the formal allocation/deallocation functions of the system, and all other code which creates or destroys accesses paths.

Identification of access management code associated with allocation/deallocation uses a technique similar to that employed in detecting attribute residuals, consisting of first identifying cell types; for each cell type identifying the corrsponding free-pool(s); and for each free-pool identifying the insertion/extraction code. The access management code is subsequently located by examining code surrounding the insertion and extraction points (in the case of allocation, code logically subsequent to the extraction point, and in the case of deallocation, code logically preceding the insertion point) for access-path creation and destruction.

In addition to the access management code in formal allocation/deallocation, access paths can be created or destroyed by any code which copies, inserts, or deletes an element of an existing access path. Much of this code can be found by a symbol search using the symbols by which access path elements are referenced. However, access path elements may not always be referenced symbolically. One way in which nonsymbolic references can occur is via offsets and pointers. Knowledge of the tables and control structures in which access paths are defined may be useful in locating relevant code. (A symbol search might be performed which uses the names of these tables as objects of the search.) The above procedure must be applied recursively so that all access path elements created by such propagation are identified along with the associated access management code.

A third way in which access paths can be created is the invocation of special operators which use the underlying translation mechanisms to produce partially or fully translated names. For example, the "LOAD-REAL-ADDRESS" operator in the IBM System/370 instruction set uses the address translation mechanism to develop a real memory address. Since invocation of such operators constitute copying of access path elements, such operators must be recognized and their invocations located.

Finally, accesses via some translation mechanisms can be interrupted (e.g., through faults) and a fully or partially translated name stored. Examination of the code associated with such interrupts is also necessary to track the use of stored interpreted names.

Having identified the access management code, the remaining task is to evaluate that code to ensure that access paths are properly created and deleted, and that all paths are properly accounted for. This evaluation process may uncover inconsistencies between the creation and destruction of access paths which may themselves suggest the existence of additional access management code.

REFERENCE

Lampson, B., "A Note on the Confinement Problem," *Communications of the ACM*, Vol. 16, No. 10, October 1973, pp. 613-615.